COMP2700 2020 SEMESTER 2 - ASSIGNMENT 1

Version 2020-08-29

SUBMISSION DEADLINE: Sunday, September 30th, 2020 - 11:55pm AET.

SUBMISSION PROCEDURE: See Wattle page for the course

1. OBJECTIVES:

The main objective of this assignment is for the students to understand the security mechanisms in Unix-like operating systems and their interplay with software security, in particular:

- to learn practical access control mechanisms in Unix operating system, such as files and folder permissions and SUID programs, and how application security is affected by them;
- to learn how software vulnerabilities can be exploited to circumvent security mechanisms in both applications and operating systems.

2. PROBLEM DESCRIPTION:

SYSTEM SETUP

This assignment is structured similarly to 'Capture the Flag' (CTF) challenges commonly used for cyber security training and education. The CTF challenges are embedded in a virtual machine (VM), containing an installation of Ubuntu 18.04 (64 bit) operating system. To do this assignment, you will need to download the Assignment 1 VM image (comp2700-assignment1.ova) for this assignment, available from the following link:

https://cloudstor.aarnet.edu.au/plus/s/5Hn8jEYcpaQGbYp

If you wish to use the RSCS Linux VDI virtual desktop (linuxvdi.anu.edu.au) to run your VM, you can also find this VM image in

```
/courses/comp2700/public/comp2700-assignment1.ova
```

in the VDI virtual desktop. (See the Lab 1 manual for instructions to access linuxvdi virtual desktop).

You need to install this assignment VM in order to do this assignment.

User accounts: There are two users in the VM: admin2700 and alice. ***For this assignment, you will solve all challenges using the user 'alice'.*** The password for 'alice' is 'alice123'. The password for admin2700 will not be provided. The intention is for you to solve the challenges without using the root account or using the admin2700 account.

THE CTF CHALLENGES FOR THIS ASSIGNMENT

There are 6 (six) CTF challenges in this assignment. Each challenge asks you to discover a 'flag' embedded in the challenge. Each flag is a string that takes the form 'flag{some_texts}' where some_texts are texts containing English words, possibly with one or more symbols (_, or punctuation symbols), unique to each challenge problem. For example, some of the flags could be something like 'flag{you_win!}'. To discover the flags you will need to find vulnerabilities in the (SUID) programs associated with the challenges, and exploit them to print the flags. The challenges are designed so that each challenge has a unique vulnerability exploitable using one or more methods we cover in this course.

You will be required to write a report documenting your exploitation process for each of these challenges and produce necessary artefacts for the assessor to replicate exactly your exploitation process. Details of the report requirements will be posted in a separate document.

GENERAL RULES FOR SOLVING THE CHALLENGES

There are specific rules for solving the challenges, but the following is a list of rules that applies to all challenges: You are not allowed to use any of the following methods to solve the challenges:

- Logging in as the root user to solve the challenges.
- Changing the system configurations of the Assignment 1 VM in any way in order to solve the challenges, e.g., by modifying the VM disks offline and give 'alice' extra privileges.
- Accessing the virtual disk of the Assignment 1 VM directly to obtain the flags, e.g., by mounting the disks in a different VM to obtain the flags.

If it is determined that your solution for a challenge uses any of the above methods, you will receive no marks for that challenge.

Below are a brief overview of the CTF challenges. Each challenge comes with a set of files (binary/source of programs or data). Not all details about these files will be provided here, so you will need to examine these files closely to learn more information about the challenges, e.g., the permissions and the owners of those files, how the program for each challenge works etc.

CTF CHALLENGE #1 - WALK/THE/PATH

Total marks: 10
Location of the challenge files in the VM: /ctf/path/

Description: There is an SUID program called 'display' in this challenge, that allows a user to display files in a certain directory. The user specifies the name of the file, and the program will append a path prefix to the file name before displaying its content.

There are only two subdirectories of interests in this case: /ctf/path/admin2700 and /ctf/path/alice. Access control is done via a check on the real user id of the process running this program; if the real uid is the same as the uid for the admin2700 user, then the program will use the prefix /ctf/path/admin2700 to add to the file name; otherwise it uses the prefix /ctf/path/alice/.

The program attempts to sanitise the input file to filter special characters (double dots) to deter an easy exploit using path traversal attack.

Your task: Find the flag embedded in embedded in /ctf/path/admin2700/. Your exploit must be related to path traversal vulnerability in the program, so exploits such as buffer overflow (if it is applicable) are not allowed.

CTF CHALLENGE #2 - I_$HELL_HACK

Total marks: 10
Location of the challenge: /ctf/shell/

Description: There is an SUID program called 'shellwrapper' in this challenge. The program when run presents several menus, e.g., to check the weather, or to read or write notes to a file (mynotes.txt).

Your task: Find the flag contained in the file 'flag.txt' in this directory. Use only the shell command injection exploit for this challenge.

CTF CHALLENGE #3 - MY MESSENGER

Total marks: 16
Location of the challenge: /ctf/message/

Description: This challenge contains an application to send and receive messages between users in a system. A user needs to create an account first. After an account is created, a file with the same name as the user (with extension .inbox) is created. Users can also import the content of another inbox. To import an inbox, name the inbox file as 'import.inbox' and put it in the home directory of the user.

Your task: There is a flag embedded in one of the inboxes in the directory of this challenge. Find that flag. You are allowed to exploit only a weakness related to file path traversal. That is there is an issue in how the program reads/writes to files. Use that weakness to read other users' inbox to find the flag.

Important Note: The location of the flag will be changed during the assessment, e.g., it may be hidden in an inbox of a user different from the inbox where the flag is currently located in the Assignment 1 VM. So make sure that your artefact submission (e.g., attack scripts) for this challenge can handle this -- see also 'Submission Requirements' below.

CTF CHALLENGE #4 - FORTUNE COOKIES

Total marks: 18

Location of the challenge: /ctf/fortune/

Description: This challenge uses a wrapper program to call a shell script that prints a line (containing cryptic messages from fortune cookies) from a text file ('fortunes.txt'). This is supposed to be a random line, but something appears broken. It always prints the same message. There is a flag embedded in one of the fortune cookies. Can you find it? This problem uses the `$RANDOM` function in shell to generate random numbers. See

https://tldp.org/LDP/abs/html/randomvar.html

for details of the use of $RANDOM function in shell.

Your task: Find the flag embedded in one of the cookies. You are allowed only to use exploits related to the shell commands and shell environments.

Important notes: The line where the secret is embedded may be changed by the assessor during the assessment process. So you need to write a script to test all possibilities where the flag could be hidden.

CTF CHALLENGE #5 - A GAME OF (RANDOM) NUMBERS

Total marks: 22
Location of the challenge: /ctf/numbers/

Description: The program is a simple loop, where a random number is generated in each iteration. The user needs to guess the random number in order to win the game, i.e., causing the program to print the flag (embedded in the file 'flag.txt'). Despite the use of random numbers, the solution does not depend on predicting the random number sequence. Look somewhere else. The stack perhaps?

Your Task: Find the flag. Use exploits related to stack-based exploits (overflow, etc).

CTF CHALLENGE #6 - LOGCAT

Total marks: 24
Location of the challenge: /ctf/logcat/

Description: This challenge contains a `logcat' program that prints either a system log (`system.log`) or a user log (`user.log`) depending on whether the real uid of the process executing the program is admin2700 (1000) or not. There is a flag embedded in `system.log`. The challenge is to capture that flag running the program as `alice`. There is an exploitable format string vulnerability somewhere in the code.

Your Task: Find the flag embedded in system.log. Use only format-string related exploits.

Important notes: The location of the flag in system.log may be changed during the assessment, so make sure your attack scripts can handle this change.

4. SUBMISSION REQUIREMENTS

You must submit a **report** detailing the exploitation process you attempt for each challenge. In addition to your report, you should provide necessary **artefacts** (programs, scripts, or other data file) necessary for the assessor to reproduce your exploits.

The requirements for the report and the marking rubrics will be provided in a separate document.

The artefact submission will be used for the assessor to judge whether your exploits actually work. It is used as an additional check to make sure you actually did successfully recover the flags (and not simply because you obtained them via other means, e.g., via root access, or copying someone else's solutions).

A submission without a report will receive 0 (zero) mark automatically irrespective of the artefacts submitted.

A submission without the artefact component will still be considered, but a penalty of up to 10% (of possible marks) for each challenge for which the artefact submission is missing, if the assessor judges that the exploit is not reproducible.

There are no specific formats for the artefact submission, but the following must be adhered to:

- The artefact must allow the assessor to reproduce your exploits in a fresh installation of the Assignment 1 VM, with possibly some modifications to the locations of the flags (as mentioned in the descriptions of some challenges above).
- The artefact must not rely on software packages or any other data not already installed in the unmodified VM.
- The artefact does not require internet connection. During the assessment, all network access will be disabled.
- You must include a README file (separate from the report), in the artefact submission, with clear and concise instructions on how to reproduce your exploits using the artefact.

All the CTF challenges in this assignment can be solved by simple shell scripts, so it is recommended that you write shell scripts to automate your exploits as part of your artefact submission. Some tips and tricks will be provided in the Discussion forum for Assignment 1 on Teams.

5. WHERE TO SUBMIT

Submissions of the solutions must be done through the Wattle site for this course. Unless explicitly approved by the lecturer, the only submission method that is allowed is through Wattle. In particular, unsolicited email submissions will be ignored.

Put your report and your artefact submission in a compressed zip file, and name it according to your first name, last name and your ANU ID. For example, if your name is John Doe, and your ANU ID is u1234567, then name your zip file John_Doe_u1234567.zip.

The submission site and the submission procedure will be provided separately on Wattle page of the course.

6. DEADLINE & PENALTY

- The submission deadline is **Wednesday, September 30th, 2020 - 11:55pm (AET)**.
- **Multiple submissions** are allowed. You can submit as many times as you want before the deadline. But only the latest submission will be graded.
- **No late submissions** are allowed without a prior approval from the course convenor.

7. ASSESSMENT

Details of the assessment rubrics for your report will be provided in a separate document.

Your artefact will be used to reproduce your exploits. Each solution to a CTF Challenge will be tested in a fresh installation of the Assignment 1 VM to ensure no interference from other solutions. A non-reproducible exploit for a challenge carries a penalty of up to 10% of the possible marks for that challenge -- even if the report correctly identifies the issues in the challenge. The exact amount of penalty will be at the discretion of the assessor. Generally the assessor has limited time budget, so exploits that demand an unreasonable amount of

work on the assessor to reproduce will be viewed unfavourably. So make sure you automate your exploits as much as possible.

For CTF Challenge #3 (My Messenger), CTF Challenge #4 (Fortune Cookies) and CTF Challenge #6 (Logcat), during the assessment, the assessor may change the location of the flag to test the robustness of your solution; see the descriptions of these challenges above for more details.

8. INTERVIEW

As a deterrence to academic misconduct, the assessor reserves the right to conduct (random or targeted) oral interviews with students to further assess students' work. An unsatisfactory outcome of an interview with a student may lead to mark reduction and/or further investigations into academic misconduct.